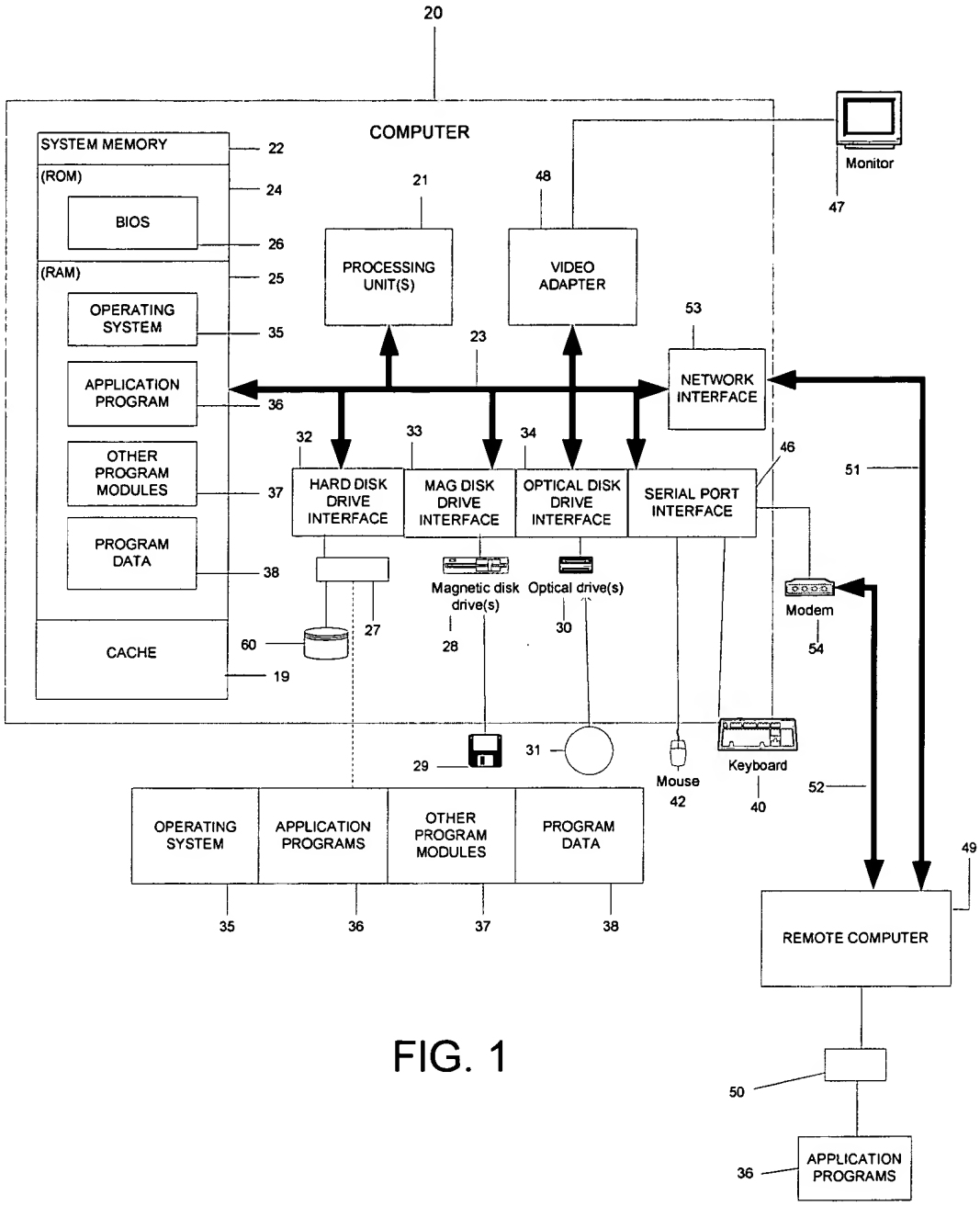
20
COMPUTER
SYSTEM MEMORY
22
(ROM)
24
BIOS
26
(RAM)
25
OPERATING SYSTEM
35
APPLICATION PROGRAM
36
OTHER PROGRAM MODULES
37
PROGRAM DATA
38
CACHE
19
21
PROCESSING UNIT(S)
48
VIDEO ADAPTER
Monitor
47
53
NETWORK INTERFACE
23
32
33
34
46
51
HARD DISK DRIVE INTERFACE
MAG DISK DRIVE INTERFACE
OPTICAL DISK DRIVE INTERFACE
SERIAL PORT INTERFACE
60
27
Magnetic disk drive(s)
28
Optical drive(s)
30
Modem
54
29
31
Mouse
42
Keyboard
40
52
OPERATING SYSTEM
APPLICATION PROGRAMS
OTHER PROGRAM MODULES
PROGRAM DATA
35
36
37
38
49
REMOTE COMPUTER
50
36
APPLICATION PROGRAMS

FIG. 1

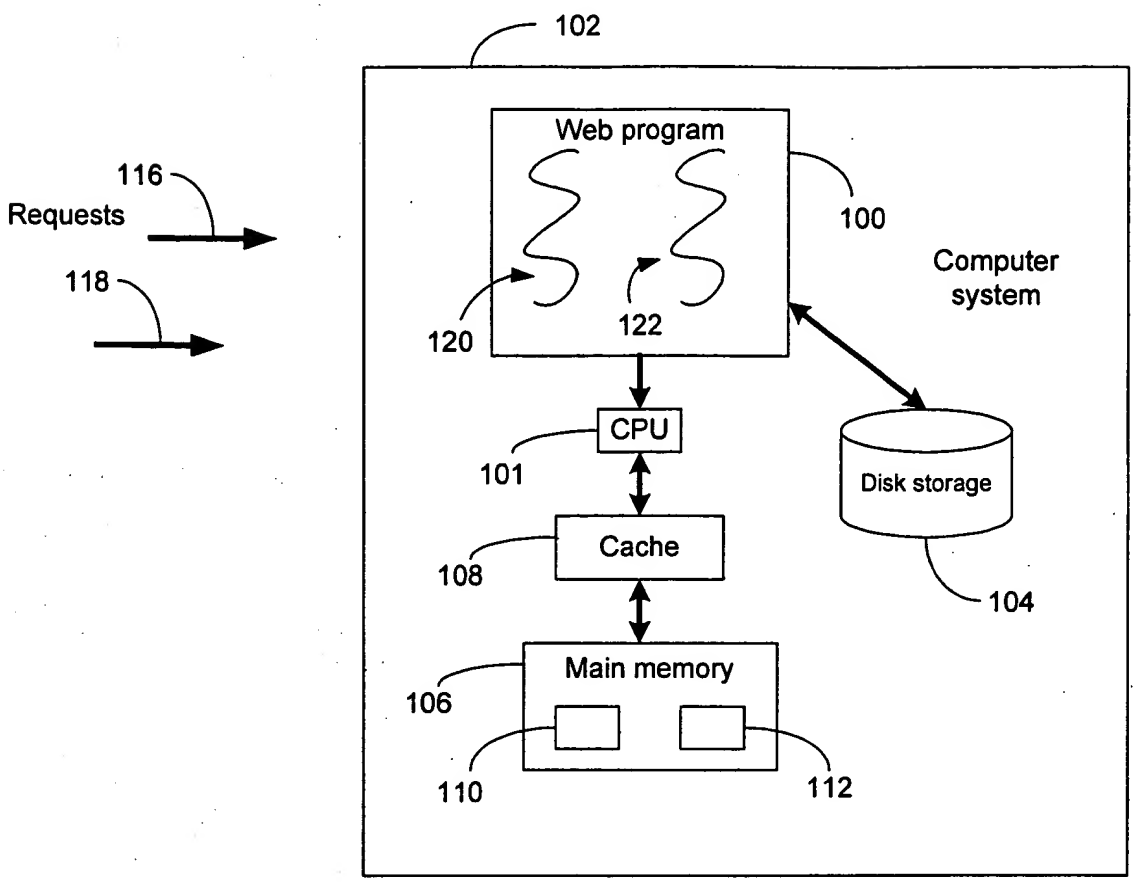
102
Web program
100
Computer system
116
Requests
118
120
122
CPU
101
Disk storage
Cache
108
104
Main memory
106
110
112

FIG. 2

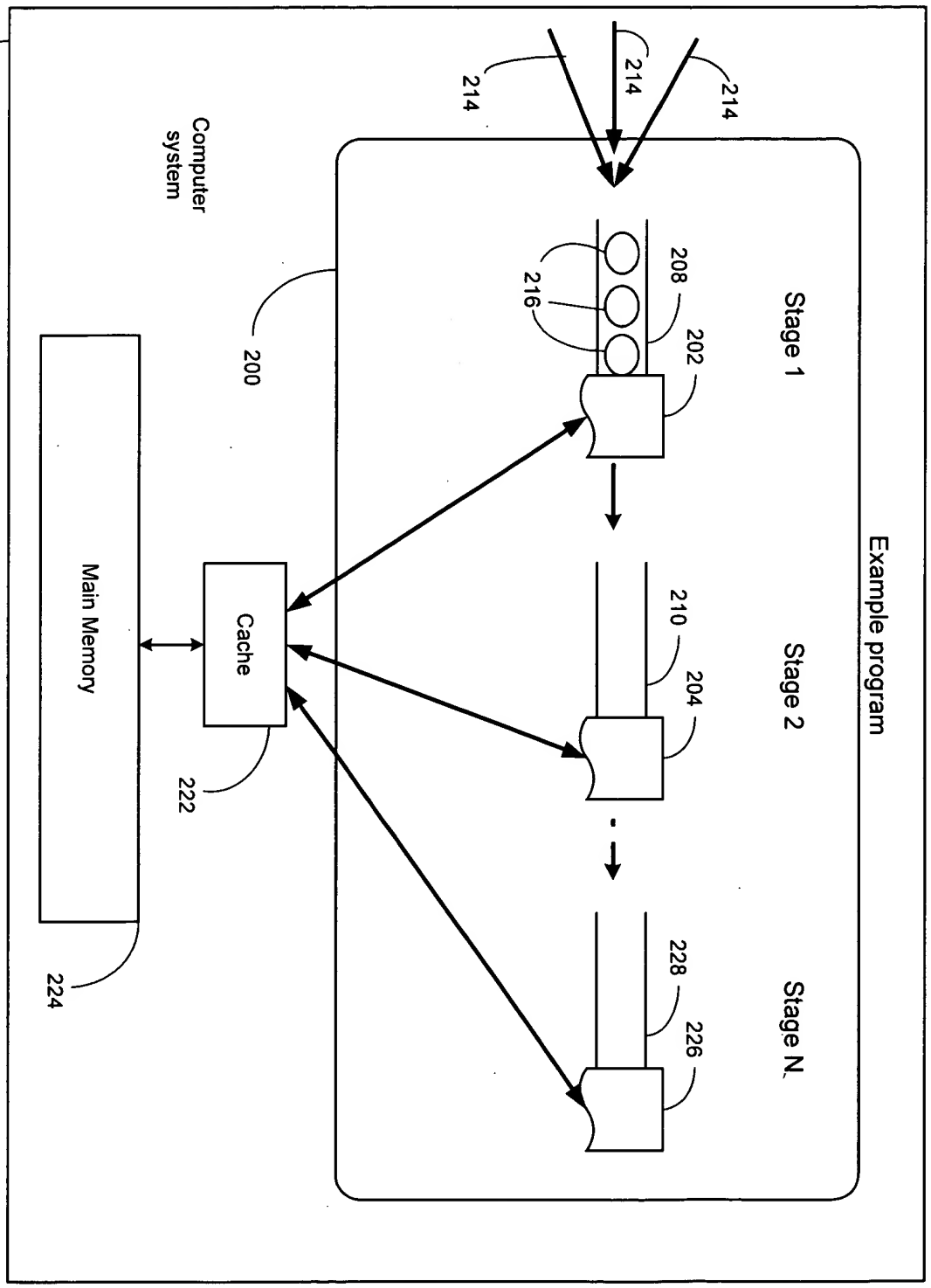

FIG. 3

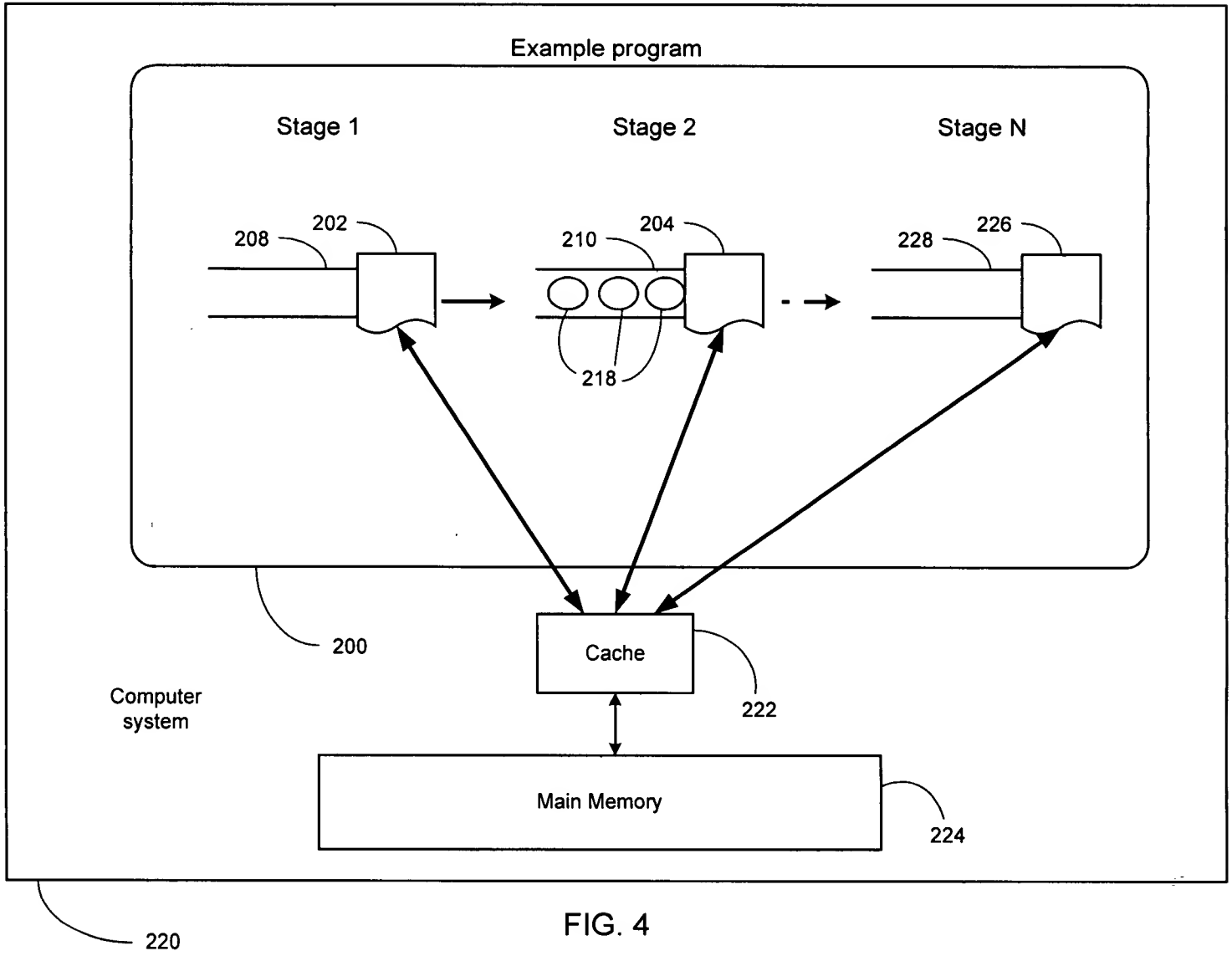

FIG. 4

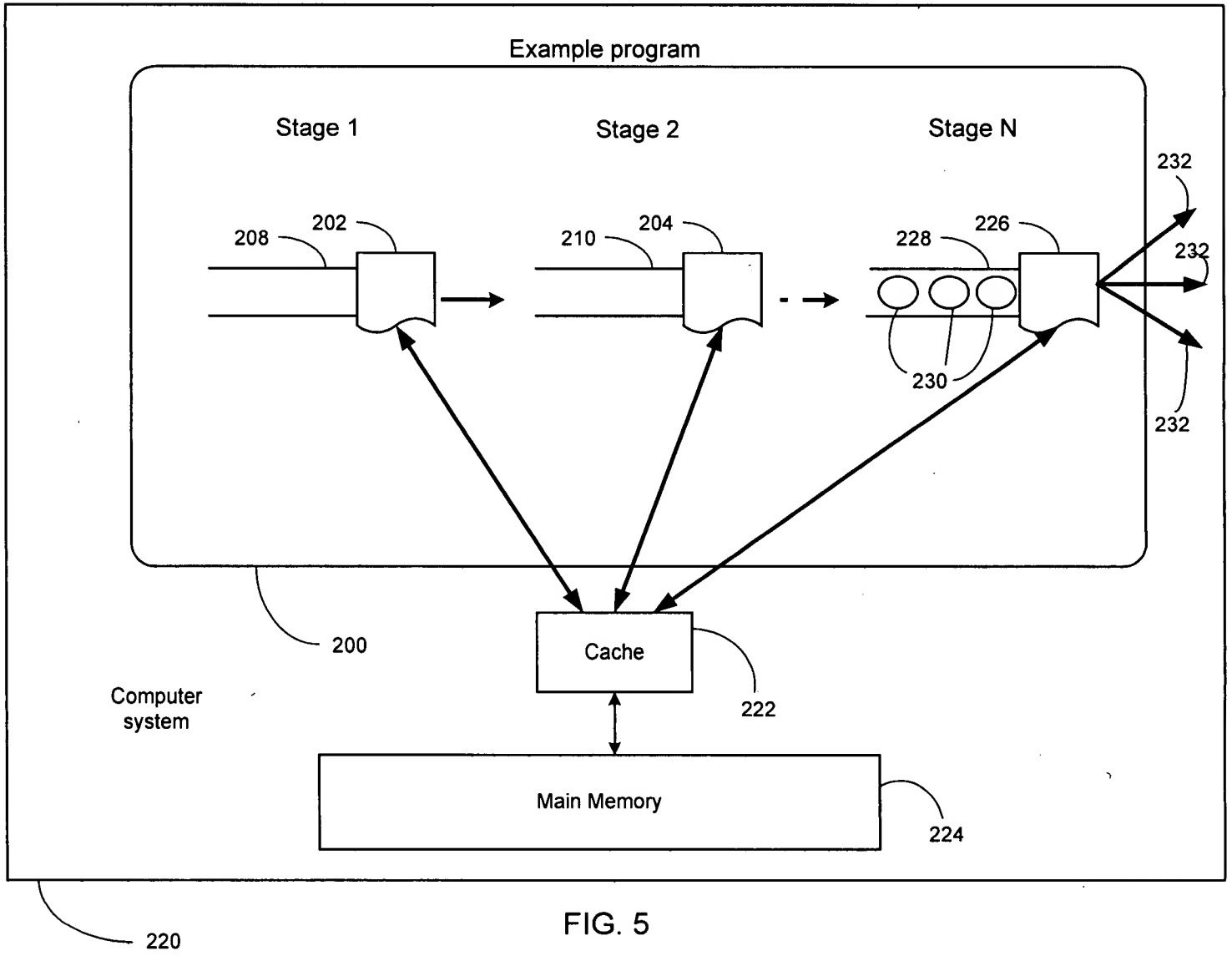

FIG. 5

Stage associated with subtask 300a

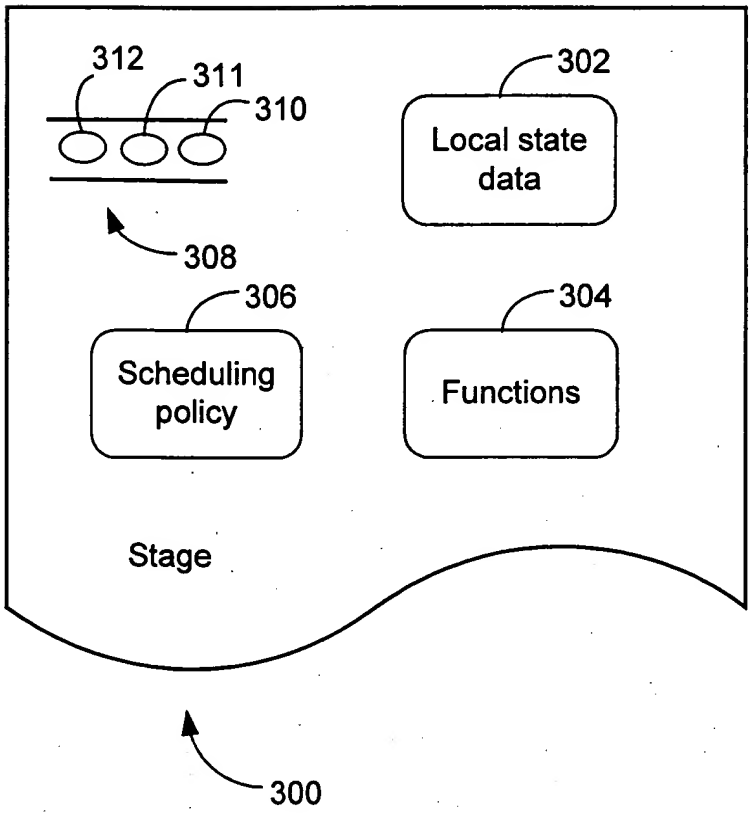

FIG. 6

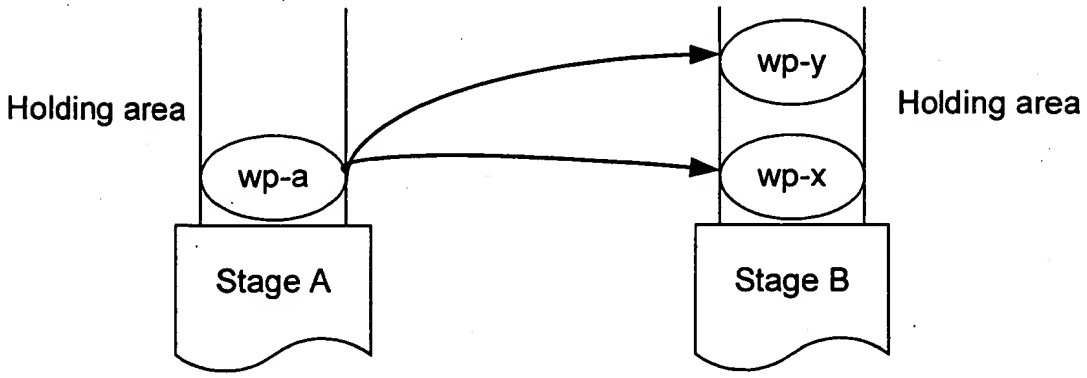

FIG. 7a

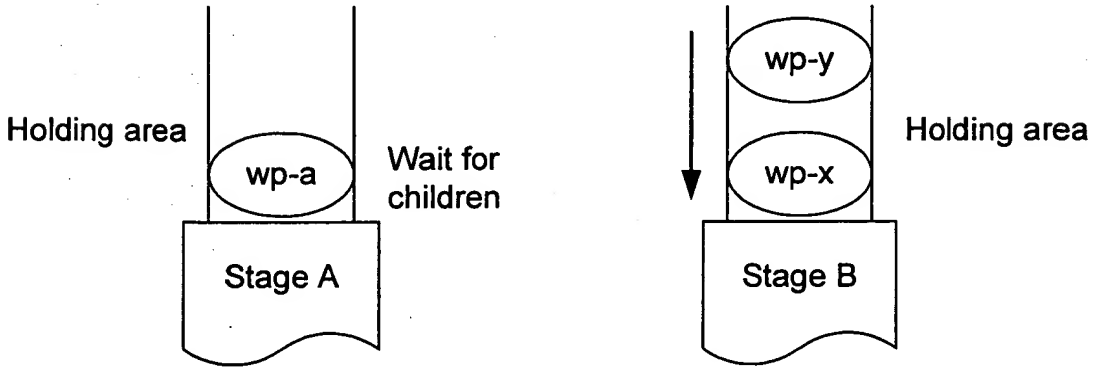

FIG. 7b

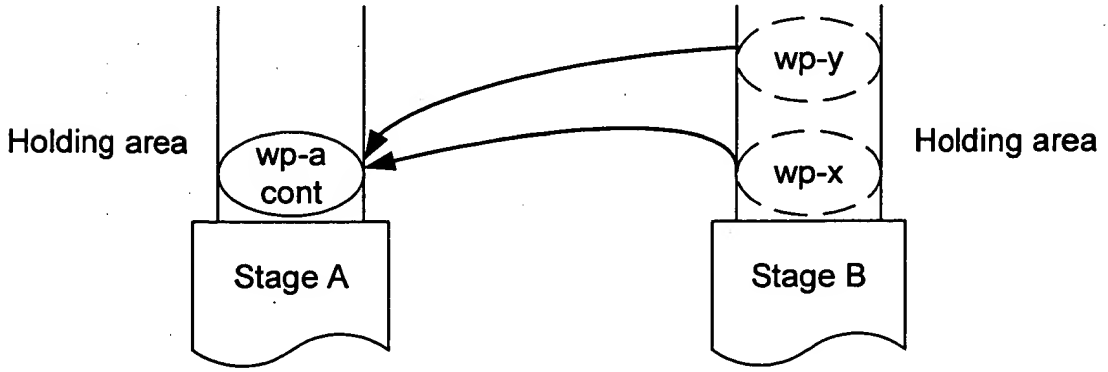

FIG. 7c

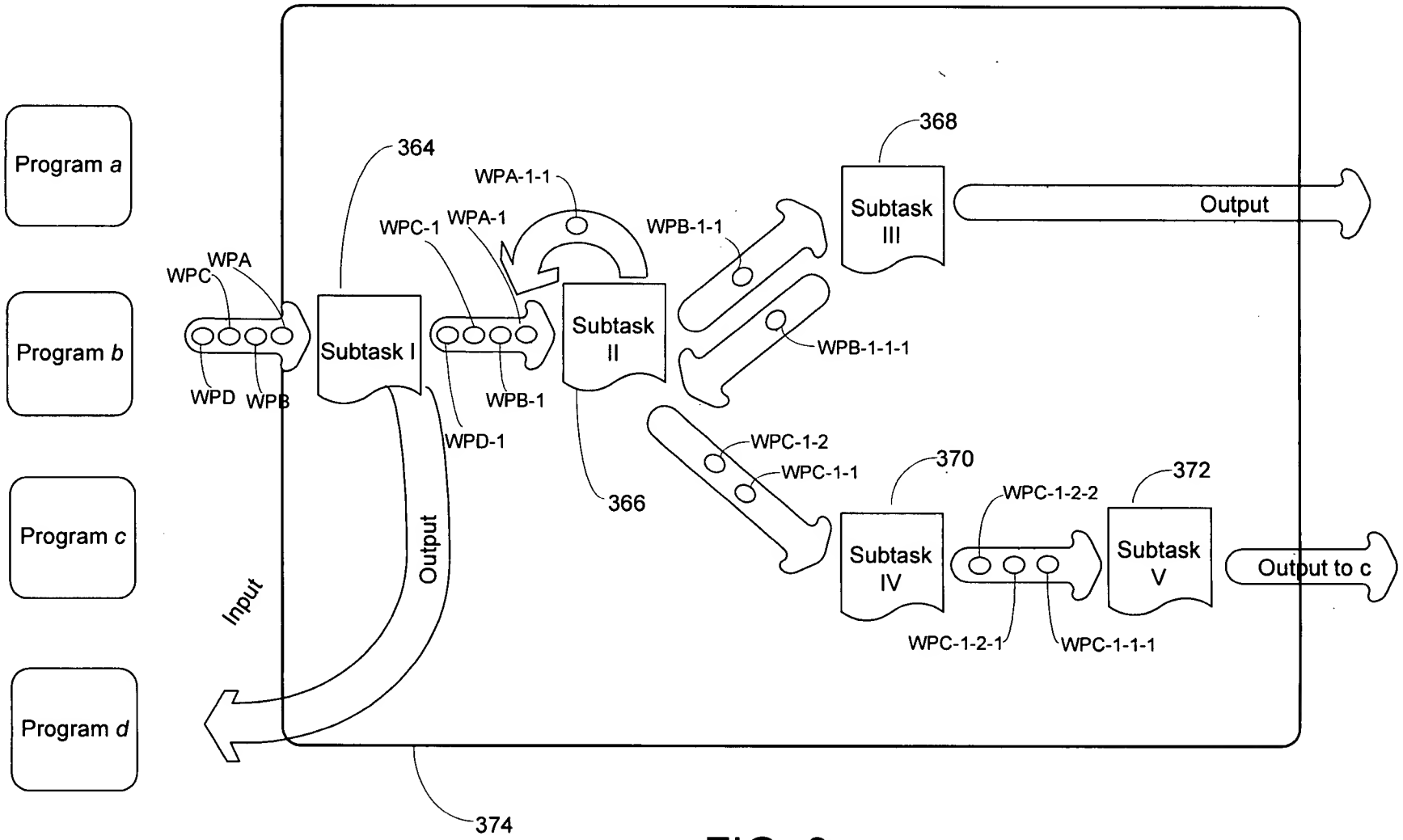

FIG. 8

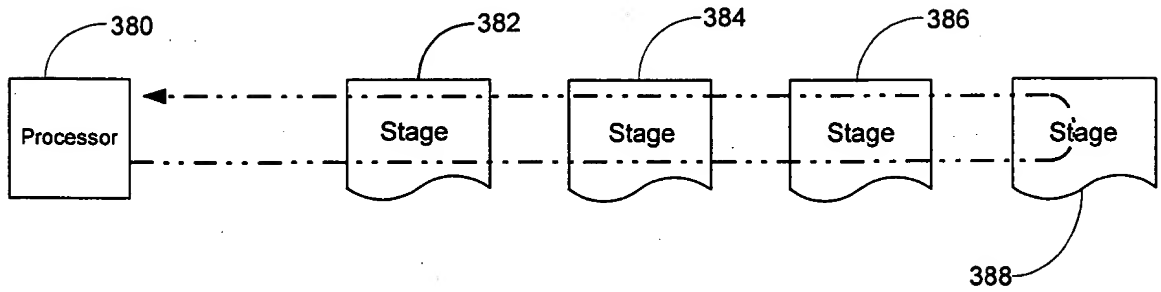

FIG. 9

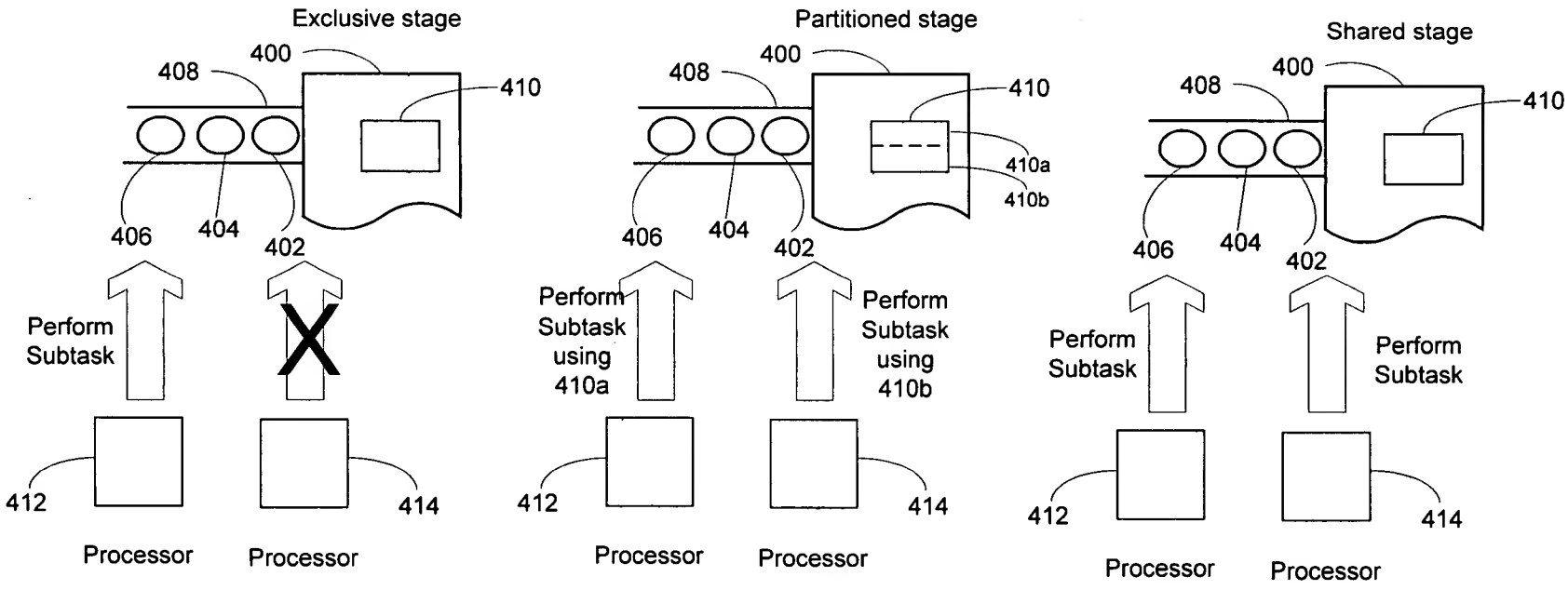

FIG. 10

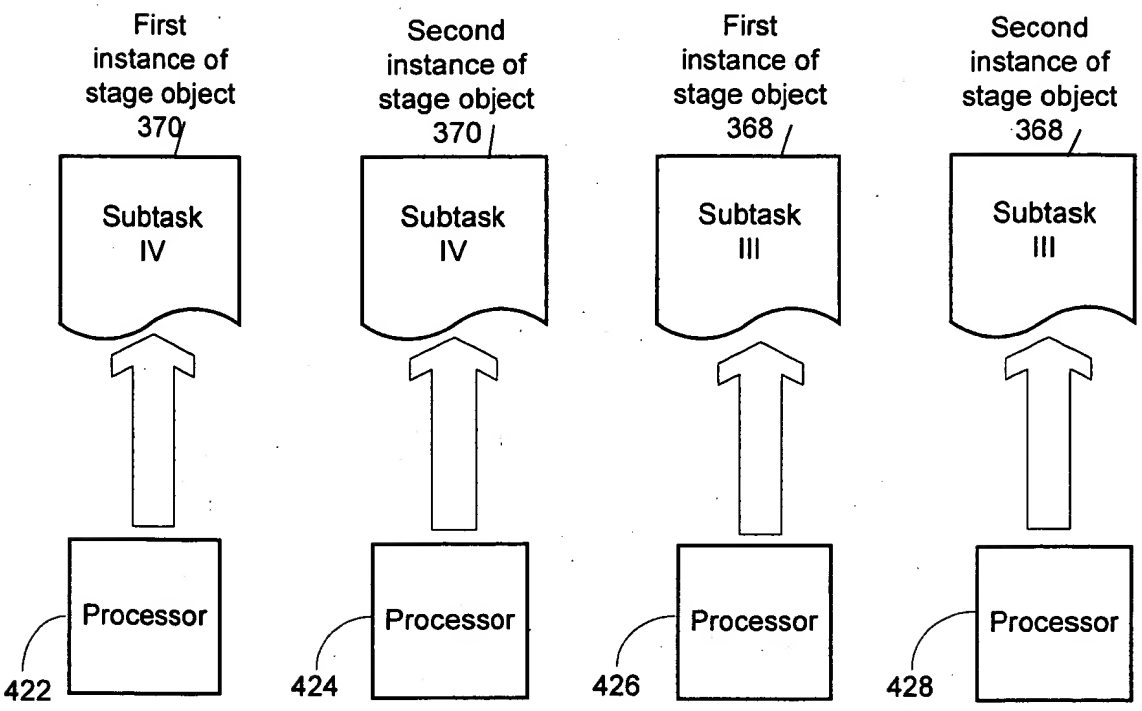
First instance of stage object 370
Second instance of stage object 370
First instance of stage object 368
Second instance of stage object 368
Subtask IV
Subtask IV
Subtask III
Subtask III
Processor
Processor
Processor
Processor
422
424
426
428

FIG. 11

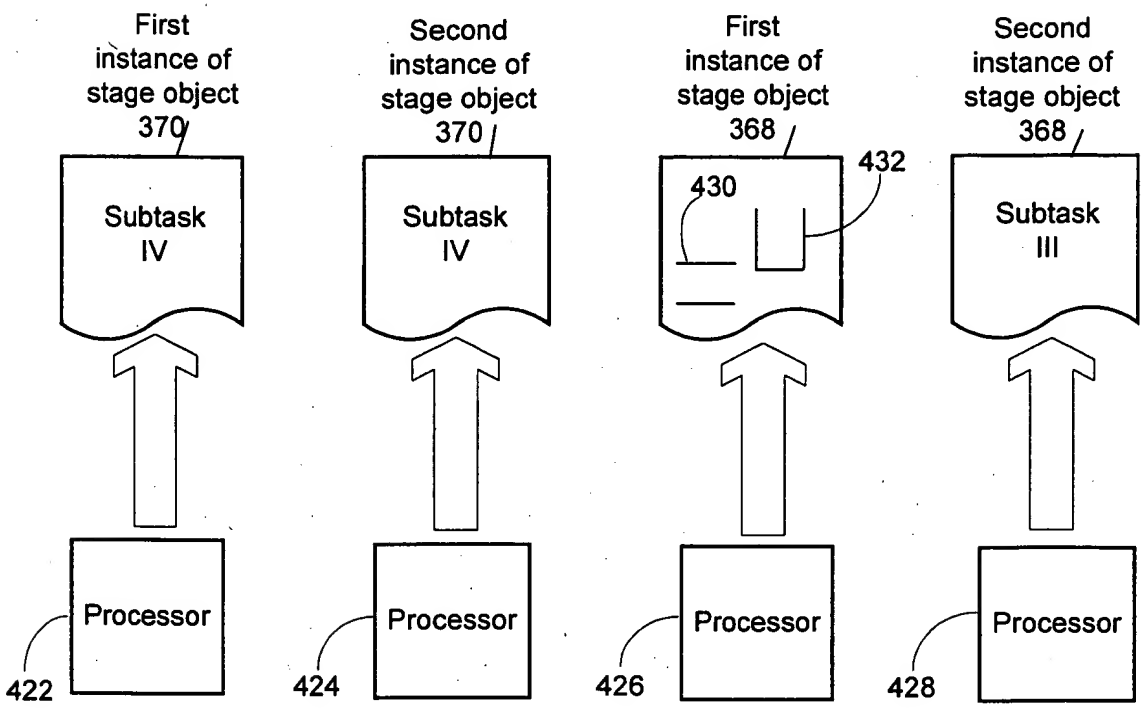
First instance of stage object 370
Second instance of stage object 370
First instance of stage object 368
Second instance of stage object 368
Subtask IV
Subtask IV
430
432
Subtask III
Processor
Processor
Processor
Processor
422
424
426
428

FIG. 12

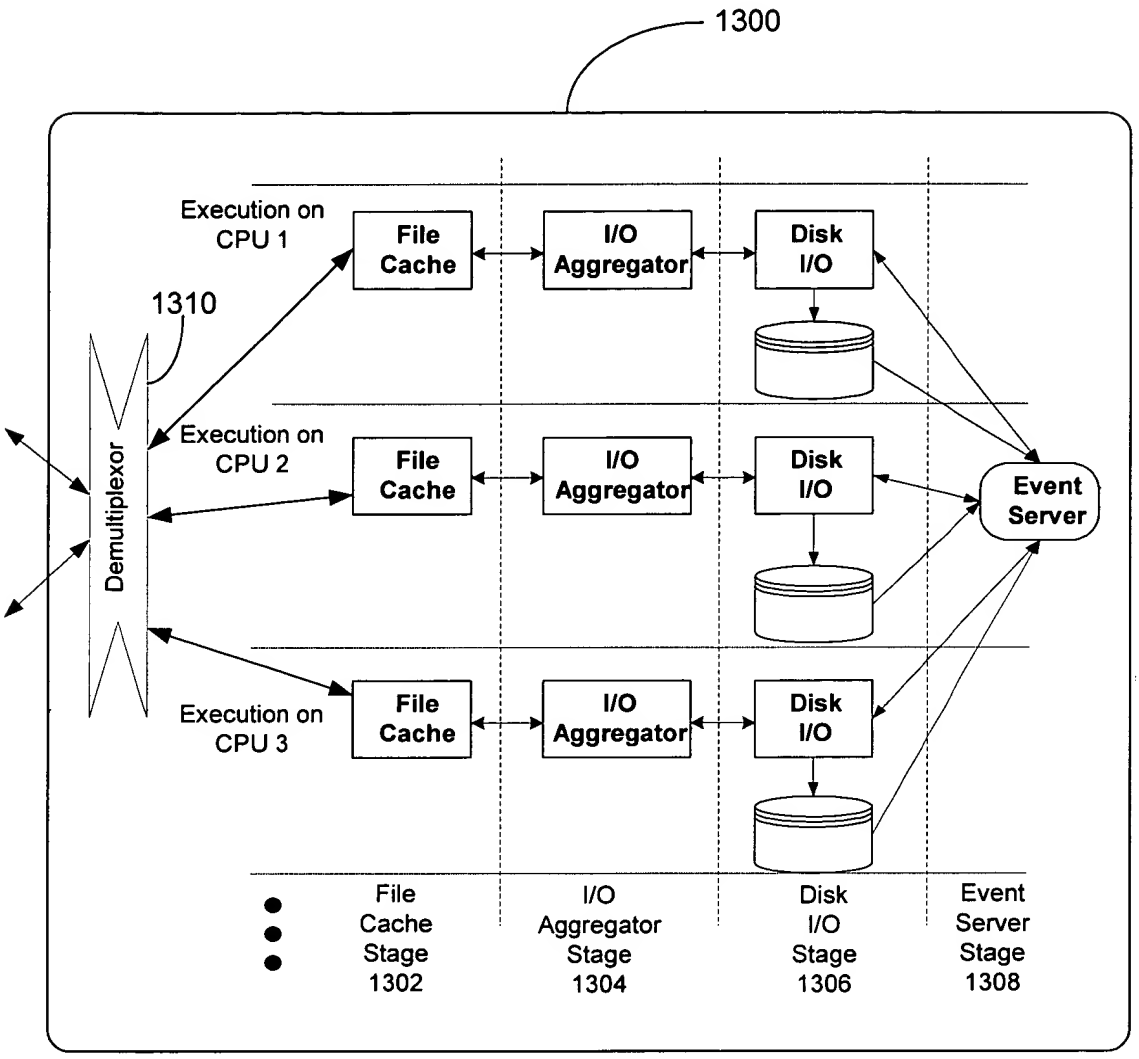
1300
Execution on CPU 1
Execution on CPU 2
Execution on CPU 3
1310
Demultiplexor
File Cache
I/O Aggregator
Disk I/O
Event Server
File Cache Stage 1302
I/O Aggregator Stage 1304
Disk I/O Stage 1306
Event Server Stage 1308

FIG. 13

```
const int MaxChildren= 4;

class MY_STAGE : public STAGE
{
public:
  MY_STAGE() :
   STAGE ("HelloWorldStage", ExclusiveStage, False, 0, 0, DefaultTimer, True, StageBatchSize)
  { }
};

MY_STAGE MyStage;          // Instance of stage used to execute operations

class MY_PACKET : public CLOSURE<MY_PACKET>
{
public:
  RESULT<INT> Children[MaxChildren];
  int Number;

  MY_PACKET() : { }

  MY_PACKET(int NewNumber) : Number(NewNumber) {printf("Creating new child %d\n",
Number);}

  // Parent Work Packet:
  ACTIONS StartChildren() {
    // Create the children and waiting for them to complete
    for (int Count=0; Count < MaxChildren; Count++) {
new(NewChild, &MyStage, NoPartitionKey, NoSessionKey, &Children[Count]) MY_PACKET(Count);
    }
    return WaitForChildren(WakeAfterSleep);
  }

  // Parent Continuation:
  ACTIONS WakeAfterSleep() {
    // Make sure all children are done
    int Total = 0;
    for (int Count=0; Count < MaxChildren; Count ++) { Total += Children[Count]; }

    printf("\nAll children have finished.\n\n");

    printf("The total of 0 + 1 + 2 + 3 = %d\n\n", Total);

    // Quit
    return Complete();
  }

  // Child Work Packet:
  ACTIONS NewChild() {
    // Code for a child, which just returns its index
    RESULT<int> Result = Number;
    printf("New child %d is now running\n", Number);
    return Complete(&Result);
  }
    };
```

FIG. 14